Cognitive feedback and behavioral feedforward automation perspectives for modeling and validation in a learning context

Gayane Sedrakyan and Monique Snoeck

Department of Decision Sciences and Information Management,
Research Center for Management Informatics, KU Leuven, Belgium
gayane.sedrakyan@kuleuven.be, monique.snoeck@kuleuven.be

(PREPRINT)

Abstract. State-of-the-art technologies have made it possible to provide a learner with immediate computer-assisted feedback by delivering ***a feedback targeting cognitive aspects of learning***, (e.g. reflecting on a result, explaining a concept, i.e. improving understanding). Fast advancement of technology has recently generated increased interest for previously non-feasible approaches for providing ***feedback based on learning behavior observations*** by exploiting different traces of learning processes stored in information systems. Such learner behavior data makes it possible to observe different aspects of learning processes in which feedback needs of learners (e.g. difficulties, engagement issues, inefficient learning processes, etc.) based on individual learning trajectories can be traced. By identifying problems earlier in a learning process it is possible to deliver individualized feedback helping learners to take control of their own learning, i.e. to become self-regulated learners, and teachers to understand individual feedback needs and/or adapt their teaching strategies. In this work we (i) propose cognitive computer-assisted feedback mechanisms using a combination of MDE based simulation augmented with automated feedback, and (ii) discuss perspectives for behavioral feedback, i.e. feedforward, that can be based on learning process analytics in the context of learning conceptual modeling. Aggregated results of our previous studies assessing the effectiveness of the proposed cognitive feedback method with respect to improved understanding on different dimensions of knowledge, as well as feasibility of behavioral feedforward automation based on learners behavior patterns, are presented. Despite our focus on conceptual modeling and specific diagrams, the principles of the approach presented in this work can be used to support educational feedback automation for a broader spectrum of diagram types beyond the scope of conceptual modeling.

Keywords: conceptual modeling, model driven development, simulation, simulation feedback, rapid prototyping, model testing / validation, feedback automation, learning process analytics, smart learning environments

adfa, p. 1, 2011.

1 Introduction

During a learning process feedback can be provided in a variety of types (e.g., verification of response accuracy, explanation of the correct answer, hints, worked examples, etc.), in a variety of forms (verbal/written text, graphics, audio, video, animation, simulation, etc.), at various times (e.g., immediately following an answer, at the end of a module, etc.) [1], by different people (e.g. teacher, peer, self,…) [2]. The role of feedback in linking learners' past and future work, and helping them to create a progressive developmental trajectory, means that ***timeliness*** should be central to any discussion of feedback [3]. Research has shown that the sooner students receive feedback the more effective it is for their learning [4]. Usually feedback is not available during a learning task completion process but is given after a certain learning task has been completed, thus having the form of ***outcome feedback***. Outcome feedback is a minimal form of external guidance, stating if an achieved solution/answer is correct or not and why. In feedback research the effectiveness of more informative types of ***feedback that can guide a learning process*** is highlighted [1, 5, 6]. Different theories have attempted to explain the *process of how people learn*. Even though psychologists and educators are not in complete agreement, most do agree that learning may be explained by a combination of two basic approaches: *cognitive theories*, i.e. ***constructivism***, that view the learning process as a step by step knowledge construction process, and *behavioral theories, i.e.* ***behaviorism***, in which learning is defined as a change of the behavior of a learner by reinforcing some aspect of his/her behavior. In the context of feedback research these approaches translate into two major forms: (1) explanations that are targeting at improving cognitive dimensions of knowledge (e.g. understanding), and (2) guidance that intend influencing a learner's behavior, e.g. engaging in a specific type of activity that is believed to be related to a successful learning path. As learning is multifaceted these approaches are often combined. For instance, in theories on ***(self-)regulation of learning*** that are closely intertwined with research on feedback and improved learning outcomes, learners are no longer viewed as repositories for information but rather they are proactive and active processors of information acting as *constructors of their own knowledge by reinforcing themselves for goal-directed behavior*. Self-regulated learning is defined as the ability of a learner to monitor and evaluate own progress with respect to self-improvement needs in the process of knowledge construction [7].

State-of-the-art technologies have made it possible to provide a learner with immediate computer-assisted feedback in the context of different learning tasks, by delivering ***a feedback targeting cognitive aspects of learning***, (e.g. reflecting on a result, explaining a concept, i.e. improving understanding). Fast advancement of technology has recently generated increased interest for previously non-feasible approaches for providing ***feedback based on learning behavior observations*** by exploiting different traces of learning processes stored in information systems (IS). Such learner behavior data makes it possible to observe different aspects of learning processes in which feedback needs of learners (e.g. difficulties, engagement issues, etc.) based on individual learning trajectories can be traced. By identifying problems earlier in a learning process it is possible to deliver individualized feedback helping learners to take control of their own learning,

i.e. to become self-regulated learners, and teachers to understand individual feedback needs and/or adapt their teaching strategies.

In this paper we present computer assisted feedback perspectives for learning conceptual modelling. We first review the general feedback needs of novices based on the challenges of learning/teaching found in the literature. Subsequently, we aim proposing computer-assisted feedback perspectives with respect to: (1) *cognitive aspects of learning processes* (concept understanding) that can address the identified challenges ("what" aspect that allows comparing current vs. good performance, i.e. what is achieved vs, what is expected), and (2) *behavioral aspects of learning* by grounding the idea of feedback on learning process analytics, more specifically by identifying learning behavior paths that can be indicative for better/worse learning outcomes ("how" aspect in terms of "how a good performance is achieved").

The proposed approach of feedback is based on the definitions of ***process-oriented cognitive feedback and behavioral feedforward*** by [8] in which the term process-oriented in the context of feedback refers to ***immediate*** feedback needs during a task completion (e.g. problem solving) process, that a learner can either be aware (learner knows whenever s/he needs a feedback) or unaware (learner does not realize that s/he needs a correction) of. We refer here to computer-assisted feedback possibilities that can be achieved before a teacher feedback can be available. Subsequently our research aims are defined as follows:

RA1: Exploring process-oriented (immediate) feedback mechanisms for addressing the learning/teaching challenges from the perspective of *cognitive aspects* of learning.

RA2: Exploring process-oriented (immediate) feedback perspectives based on *behavioral characteristics* of novices' learning processes that, in addition to being directed to a learner, can also help a teacher to observe learning processes and based on identified inefficient processes to also rethink/adapt instructional materials/processes.

The cognitive feedback is achieved by a combination of MDE-based simulation and automated feedback. The implications for behavioral feedforward perspectives are further discussed based on the findings of our previous research proposing adopting process analytics view on learning modeling [8-10].

2 Reviewing cognitive feedback needs through the prism of learning challenges

While experienced requirements engineers and business analysts manage to mentally picture the prospective system in their mind when transforming requirements into formal conceptual models, such ability to truly understand the consequences of modeling choices can only be achieved through extensive ***experience***. However, the tacit knowledge experts have developed over time is difficult to transfer to junior analysts. While teaching such knowledge and skills to junior analysts is already a challenging task considering that system analysis is by nature an ***inexact field of science***, transferring the academic knowledge and skills to real world businesses is yet another concern as the classroom and real world situations are not identical. In their early career the

error-prone problem-solving patterns of juniors lead to incomplete, inaccurate, ambiguous, and/or incorrect specifications [11, 12]. When detected later in the engineering process such requirements and modeling errors can be expensive and time-consuming to resolve. This significant gap between the knowledge and skills of novices and experts triggers the question of *how analysis and modeling skills can be trained to facilitate the fast progression of novice analysts into advanced levels of expertise*. Amongst the factors affecting modeling process quality and learning outcomes of novices are:

- *Lack of comprehension methodologies: Understandability* (a model's ability to be easily understood) has been extensively evaluated in the literature both for static and dynamic aspects of modeling pointing out to comprehension difficulties both by practitioners and juniors due to the lack of comprehension methodologies [13];
- *The cognitive aspects of modeling*: Studies on comparing model quality checking approaches of novices and experts indicate the poorly adapted cognitive schemata of novice modelers to identify relevant triggers for verifying the quality of models [11];
- *The complexity of modeling tools*: being too "noisy" with various concepts, which can result in misusing concepts and creation of unintended models [13, 14] thus making them less effective in supporting a teaching process [15];
- *Lack of understanding of domain requirements*: Students have a hard time for achieving a thorough understanding of a set of given requirements. Absence of intensive trial and error rehearsals in the classroom [11] and the lack of possibilities to interview stakeholders in a requirements gathering process are considered the major source of limitation in novices modeling experience;
- *The lack of validation procedures and tool support*: In addition, the lack of established validation procedures [16] makes the conceptual modeling for novices very difficult to learn.
- Additionally, several researchers correlated novices learning achievements in modeling with the *lack of technical insights* considering the absence of technical components (such as computer-assisted learning) from education as a major contributing factor to the lack of preparedness of their skills [17]. Furthermore, there are aspects that cannot be obtained through reading and lecturing alone, e.g. the *dynamic representation of a system*.

3 Simulation as a cognitive feedback

Cognitive feedback gives information to learners about their success or failure concerning the task at hand provided through prompts, cues, questions, etc. that helps learners reflect on the quality of the problem solving processes and solutions so that they construct more effective cognitive schemas to improve future performance. Cognitive feedback targets at improved understanding of intermediate solutions of a learner allowing improving a problem solving process and its outcomes[18]. Simulation is known to be an excellent technique allowing understanding complex structures and behaviors and has been successfully used in a variety of learning domains, such as science

education [19], mining engineering [20], aerospace engineering [21], biological engineering [22], etc. Among key education benefits of simulation is the ability to provide *feedback* and *deliberate practice* [23]. The interventions of simulation in a learning process can be described as an ability to produce ***externally observable outcomes*** that can trigger ***internal feedback*** engaging ***self-regulatory learning*** mechanisms of learning (e.g. *self-assessing of own performance with respect to the expected performance* in terms of capability of achieving a satisfactory quality of a prospective system, *identifying needs for improvement* and *adapting* in terms of engaging in further trial/error activities for adapting a model of a prospective system that served an input for simulation). The externally observable outcomes in the form of simulation serve as a *cognitive feedback* in terms of improving understanding of a problem by reflecting on intermediate solutions of learners during a learning process and as such are also *learning process oriented*. Simulation is also known to allow *skill acquisition that accompany knowledge*. Some skills follow from conceptual knowledge whereas others involve intricate activities to develop, i.e. ***experience*** [24]. Thanks to realistic scenarios and equipment, simulation allows for expertise training through *deliberate, repetitive and evidence-based practice* (e.g. retraining till one can master the procedure or skill) [23] that is also coupled with cognitive feedback. Achieving cognitive process-oriented feedback through simulation requires:

- designing and building an simulation instrument,
- ensuring its support for the intended goal as a cognitive feedback,
- ensuring its support for the intended goal as a process-oriented feedback.

In the context of conceptual modeling learning achievements can be measured by the capability of producing physical models with high ***semantic quality***, i.e. the level to which the statements in a model reflect the real world in a valid and complete way [25]. In order to check a model for validity, a person needs to read and ***understand the model*** and compare his/her understanding of the model with his/her understanding of the given domain description. On the knowledge side, this requires an appropriate level of *modeling knowledge*, *modeling language knowledge* (e.g. understanding the modeling concepts, graphical notation) and *domain knowledge* among others [26]. To our knowledge, no research can be found in the context of courses that use simulation of object-oriented multi-view conceptual models (i.e. combining structural and behavioral aspects), nor empirically proven learning benefits have been reported for a certain simulation tool. The reason is that the existing standards for simulation technologies also introduce a number of shortcomings. The major disadvantages include:

- simulation is too complex and time consuming to achieve by novice modelers whose technical expertise is limited.
- it is sometimes difficult to interpret the simulation results.

Among different *types of simulation* (symbolic or graphical animation, execution, prototyping), the ***method of prototyping*** is capable of achieving the ***most concrete form of a prospective system***. Semantic prototyping method and tool was introduced by [27] with the goal to improve conceptual model comprehensibility however aiming at facilitating communication with stakeholders rather than a support for learning. Among the

variety of forms of prototypes in this research we refer to the definition of a prototype as "fully *functional* to prove a concept" [28]. This goal is achieved through the creation of an experimental full-scale working exemplar of a model that illustrates the typical qualities of the prospective system based on the design of its model. Prototyping is also thought of as a type of design language [29]. The learning context of prototyping as a design language includes testing of a function of a prototype with the purpose to identify potential issues concerned with *problem understanding with respect to its design* [28]. We will therefore use the terms "simulated model" and "prototype" interchangeably.

3.1 MDE-based simulation for conceptual modeling: CodeGen

We followed the principles of *Design Science in Information Systems research* which proposes two main guidelines 1. *building* and 2. *(re)evaluating novel artefacts* to help understanding and solving knowledge problems [30]. In this work we refer to simulation of a conceptual model as a process of generating prototype applications using a conceptual model as input. The Model-driven architecture (MDA) is the collection of current OMG standards for model-driven engineering (MDE), enabling, among others, code generation. MDA allows designing ***platform independent models (PIM)*** as the main representation of a system-to-be that have a sufficient level of completeness to generate other models or code from them; MDE focuses on ***transformation(s)*** (mappings) from platform independent to platform specific models or code, a process that may pass through a number of mappings before a software artefact can be generated. However, existing MDA/MDE solutions require extensive training due to the large set of skills required for using accepted standard MDA/MDE technologies, such as Unified Modeling Language (UML). As stated in [31]: "The technical complexity of UML has been held responsible for modeling adoption issues. Few expert modelers can rapidly evolve an application from requirements to code. Many of today's modelers are casual in their approach; MDA, however, requires increased rigor and training in UML modeling". Among the other fundamental deficiencies of UML is that it is unclear how to combine interactive, structural and behavioral aspects together in a single model [32]. The same holds true for the OMG's MOF and XMI standards which are used to store, transport and exchange models between tools, that are also associated with issues like semantic mismatches, version incompatibilities (XMI/UML/MOF), human-readability, etc., e.g. [33-36]. The new standards providing key technology for expressing application domains in a platform independent manner that are in addition executable include executable UML (xUML), foundational UML (fUML) - an executable UML standard that specifies precise semantics for an executable subset of UML, and Action language for fUML (Alf) - an executable UML standard that specifies a textual action language with fUML semantics. These however do not bring the MDE any closer to the novice modelers or simplify it such as making model validation by means of rapid prototyping easily feasible for business domain experts who lack technical expertise. Still a very detailed diagramming with fUML is required and a solid knowledge of both fUML and Alf is required to make further transformation of UML to code. Thus, the practical utility of MDE is still limited by the fact that:

- UML lacks a methodology to achieve a right design within a short time to be further processed with an MDA/MDE approach.
- MDE model-to-model and model-to-code transformations are hard to write, trace/debug, maintain and reuse.

In this research to achieve an in-house prototyping solution, i.e. designed and implemented rather than relying on third party code generation tools [8], we rely on the MERODE methodology [37], as the benefits of models designed in MERODE specific JMermaid environment include:

- Starting from a high-level PIM (close to a Computational Independent Model (CIM)) allows removing or hiding details irrelevant for a conceptual modeling view.
- It relies on a ***domain specific language*** and proprietary MERODE modeling environment that uses a restricted part of UML adapted to conceptual modeling goals.
- It provides a framework for ***combining structural and behavioral views*** into a single model using two prominent UML diagrams – 1. a class diagram, 2.statecharts, and a CRUD-matrix based collaboration view called Object-Event Table (OET) that defines how statecharts interact (see Figure 1).
- It allows achieving ***executable PIM*** that have a sufficient level of abstraction, while being sufficiently complete to enable applying transformation(s) from platform independent to platform specific models or code.

In the learning context of prototyping-based teaching, this research builds on, and tackles the issues of the experiences from the first iteration of conceptual model prototyping. The first version of the prototyping environment was achieved by means of the AndroMDA open source code generation tool combining its existing XMI-based cartridges and a MERODE-specific cartridge that provides an XMI transformation from a model designed in a MERODE environment and specifies a functionality of generated prototypes such as basic interface consisting of buttons, corresponding input windows they trigger and event handling mechanisms that ensure the functionality [39]. The tool already demonstrated certain positive effects in a learning context (with student evaluations of the usefulness of the tool from two academic years resulting on average 3.46 and 3.7 in the range of 5-point Likert scale). However, despite its merits, a number of usability issues negatively impacted the intended utility in the learning context, among them being time-consuming in terms of requiring multiple steps to achieve and launch a generated prototype, and issues with the intuitiveness of the user interfaces to support easy navigation and

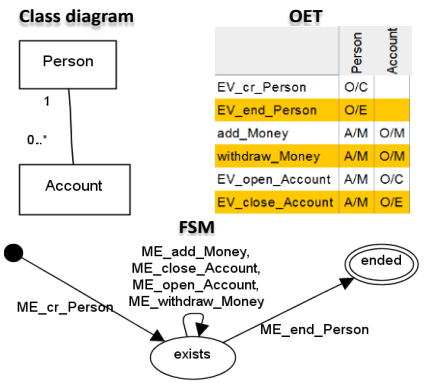

Figure 1: Modeling views within MERODE modeling environment: class diagram, Object-Event Table (OET) and a Finite State Machine (FSM) [38]

testing, e.g. it was not clear how the prototype links to a model, making it difficult to apply the method in a teaching/learning context. As a result, the evaluation survey revealed that the majority of students seemed to be reluctant in using the tool in their learning process resulting mostly in the "didn't use" answers while assessing the tool. In this paper an in-house prototyping method is introduced based on a ***template-based transformation*** [8] going straight from model to code (i.e. a model-to-text transformation) allowing to generate a prototype with a ***single click*** [8, 38, 40]. Such instant prototype production serves as a quick simulation technique that raises the usability as it lowers the required skill-set for its use and allows verifying the conformance of conceptual designs and the description of the domain in a fast and easy way. By enabling a fully functional output the method also serves as a rapid prototyping and simulation instrument. This allows assessing the generated prototype (simulation results) with respect to the expected outcome. In case of a semantic mismatch the desired outcome can be achieved through a trial and error correction process by means of modification, regeneration and verification loops.

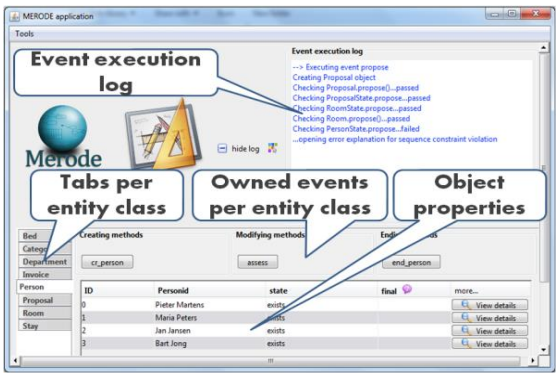

Figure 2: The main GUI of the prototype application [38].

Such an approach yields additional benefits such as better support for process-oriented assistance allowing developing modeling competences by engaging a learner in a "trial and error learning process" [41], test the incrementally modified (growing) prototypes and letting him/her check the semantic conformance of a model with the domain description. In addition, user interfaces were adapted to support maximally intuitive user experience. A user interacts with the generated application through the graphical user interface (GUI) which offers basic functionality like triggering the creating and ending of objects, and triggering other business events. Figure 2 shows the main interface of a generated prototype.

3.2 Enhancing simulation with feedback to facilitate interpretation of simulation results

It is known that simulation accompanied with feedback can result in better learning outcomes [17, 18, 22, 42]. More commonly, human instructors provide feedback for simulations usually with a post-simulation debriefing [46]. However, feedback automation methods that can guide a learning process with simulation are to our knowledge absent. In this research we present a feedback automation method that embeds a feedback generation mechanism into a simulation of a model thus allowing achieving feedback-enabled simulation. We make use of ***negative corrective feedback*** [47, 48] based on two type of formats: (1) ***textual explanations*** of the causes for the errors (execution

failures as a result of constraint violations) that explain the involved modeling constructs and their implications with respect to execution outcomes [49] and (2) improved transparency between a prototype and its model by means of ***graphical visualization*** that links the execution results to their causes in the model [49]. The inclusion of textual/visual feedback into simulation is achieved by the generation of feedback as a response to execution failures in a prototype application targeting a facilitation of interpretation of testing (simulation) results. The errors include event execution failures that result from constraint violations, which are regarded as invalid actions from the domain perspective. The goal of the incorporated feedback in the simulation loop is to facilitate the process of verification of semantic validity of the model allowing to detect errors in a model's design.

3.3 What is needed to set up an automated simulation feedback ?

In this chapter we present the architectural design of the automated feedback approach [49]. Thereto we identify assessed by comparing two such sets, goals being completeness and validity. For semantic quality, completeness is achieved if the physical representation (the model) contains all the statements of the domain, and validity is achieved if what is true or false according to the model is respectively also true or false according to the domain rules. Model simulation can be used to assess model completeness by simply verifying the presence of desired functionality in the prototype. the model elements used to set up an automated feedback. According to [26], in the conceptual modeling quality framework each framework element can be considered as a set of statements. Model quality is assessed by comparing two such sets, goals being completeness and validity. For semantic quality, completeness is achieved if the physical representation (the model) contains all the statements of the domain, and validity is achieved if what is true or false according to the model is respectively also true or false according to the domain rules. Model simulation can be used to assess model completeness by simply verifying the presence of desired functionality in the prototype. Assessing the validity of the model requires verifying the truthfulness of a statement in the prototype. In other words, if something should be allowed according to domain rules, then this should be allowed according to the model as well, and if something is forbidden according to domain rules, then a corresponding constraint should be included in the model. To verify validity, a modeler needs to define test scenarios and define an oracle (desired outcome) for each scenario according to the domain rules. The results of the execution of the test scenario are compared to the oracle to determine the semantic correspondence between model and domain. While novice

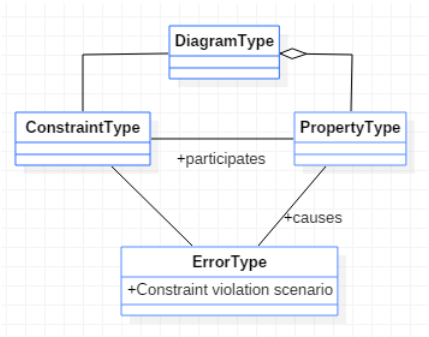

Figure 3: Model-elements used for a feedback [49]

modelers seem at ease with using a fast prototyping approach for the verification of model completeness, we witnessed that novice modelers have difficulties in understanding why a test scenario fails and relating the cause of the failure to model constructs.

Table 1: Examples of model elements used to construct feedback for class diagram and statecharts [49]

Diagram	Constraint type	Error type	Explanation & model properties
Class diagram	Cardinality of minimum 1	Create-event execution failure	an *object of type A* is attempted to be *created* without choosing an *object of type B it is associated with*
	Cardinality of maximum 1	Create-event execution failure	an *object of type A* is attempted to be *created* for which an *object of type B associated with a cardinality of max 1* is chosen which already has been assigned *another instance of an object of type A*
	Referential integrity for creation dependency	Create-event execution failure	an *object* is attempted to be *created* before the *objects it refers to* were created
	Referential integrity for restricted delete	End-event execution failure	an *object* is attempted to be *ended* before its *"living" referring objects* (objects that did not reach the final state of their lifecycle) are ended
Statechart diagram	Sequence constraint	Event execution failure	an *event* is attempted to be executed for an *object* whose *state* does not enable a *transition for that event*

Test scenario failure finds its origins in constraint violation. For example, if a course can be attributed to at most one teacher, then assigning a second teacher to a course will result in a constraint violation and a failed test scenario. Therefore, the first step in our architectural design includes the identification of the *constraints* that are supported by a *diagram type*. Next, the typology of errors with respect to the *constraint types* are specified. We also need to identify the *diagram properties* that take part in those constraints. The *error type* can be described as a constraint violation scenario. The error type contains a reference to the violated constraint type and also encapsulates the properties that *participate* in the context of the event execution and those that *cause* the error (execution failure). Figure 3 depicts the generic meta-model on how error types are related to the corresponding model elements. As mentioned earlier in this paper we

realize our approach in the context of one specific type of models, namely, conceptual models, that combine structural and behavioral aspects of a system. The modeling approach uses a combination of a class diagram (to realize the structural aspects) and multiple interacting statecharts (to support a system's dynamics). In the class diagram, constraints are captured as cardinality constraints (mandatory one, maximum one) and referential integrity constraints (creation dependency and restricted delete). In the case of a statechart, constraints are captured as sequence constraints. For each of these constraints, a corresponding error type and explanations used for feedback can be constructed as shown in Table 1. Explanations include model properties (underlined in column "Explanation & model properties").

3.4 How the approach can be realized: inclusion and generation of simulation feedback

The feedback generation mechanism is handled by inclusion of a feedback generation package in the output of the model-to-code transformation and is illustrated by the conceptual model shown in Figure 4. This package is responsible for 1. capturing the execution errors (failures) and mapping them with corresponding causes; 2. identifying the causing model properties as well as those being involved/affected; 3. matching the causes with relevant feedback template for a textual feedback; 4. generating feedback dialogs with the textual explanation and 5. further extending the textual explanation with its graphical visualization.

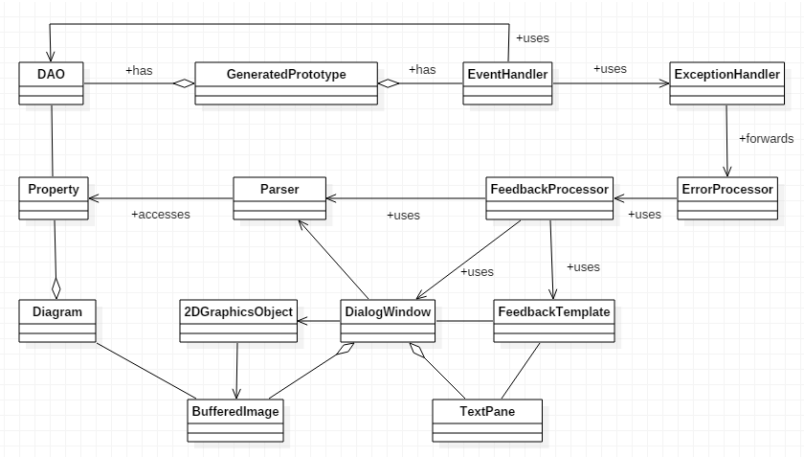

Figure 4: Architecture of the feedback generation in the context of MDE-based simulation [49]

In the model-to-code transformation the event execution process is supported by the event handler which is responsible for the transaction logic specified by a model. Error messages are generated in case of failed precondition checks. The model-to-code transformation is presented in our previous work [38] and, as it is not the core subject of this

paper, the transformation process therefore will not be covered in detail. We will however refer to some aspects of the model-to-code transformations that are relevant for feedback generation. This includes the notion of a parser and Data Access Objects (DAO) in the generated transformation. DAOs provide a simplified access to model properties stored in a database layer of the transformed code (e.g. key-value maps containing a collection of object properties such as a name, collections of attributes, events, dependencies, states, etc.) which are also used for feedback purposes.

```
feedbackText = "The FSM (statechart) of " + objectName + " puts a constraint on "
             + transitionEventName + ". The current state of " + objectName
             + " is " + objectStateName + ". In the state " + objectStateName
             + " there is no transition enabled for the business event "
             + transitionEventName + ". Look at the FSM to find which "
             + "business events are allowed in this state or find the "
             + " state(s) at which you can execute the business event "
             + transitionEventName + ".";
```

Figure 5: Sample textual feedback template for a sequence constraint violation [49]

```
feedbackText = "You already have one instance of "
             + objectType + " and according to the " + diagramType
             + " you cannot create a second instance of "
             + objectType + " because of the cardinality constraint"
             + " of maximum 1.";
```

Figure 6: Sample textual feedback template for a cardinality constraint violation [49]

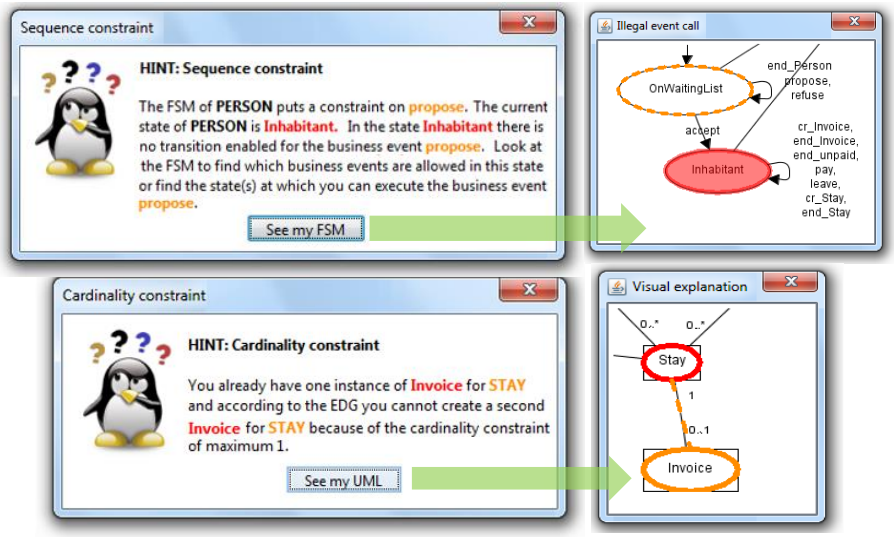

Figure 7: Sample generated textual and graphical feedback for a UML class diagram and a finite state machine (FSM) [49]

These properties are constructed during the transformation process using a parser and Apache Velocity Templates and are accessible in the final code. In the generated application the execution failures are implemented as exceptions. The exception handler contains the cause of the exception such as a reference to the corresponding constraint type along with the model properties involved in the constraint violation in a lightweight data-interchange format (comma separated string). The exception handler identifies the exception type and in case a model related execution failure is detected (there can be code related exceptions too) further links to the corresponding error processor responsible for model related errors. The error processor further derives the necessary properties error message data stream, converts them into appropriate formats and forwards to the feedback processor. The feedback processor uses a feedback template to provide a textual explanation on the corresponding parts of the diagram along with the properties of a diagram causing the execution failure as well as those being involved/affected. Sample textual feedback templates are presented in Figure 5 and Figure 6. Using the model parser the coordinates of model properties from the GUI model of a diagram are passed to a 2D graphics object. The parser is used to access any other model properties that are required to provide a hint for a possible correction scenario (e.g. if an event execution fails due to an object state, the state(s) in which the execution is allowed are used to construct a hint). The 2D graphics object is used to access the coordinate, color and font management system of the buffered image (an image with an accessible buffer of image data) of a diagram. This allows to highlight the parts of the diagram that contains the constraint that causes the error as well as to visualize the suggested hints for the correction of the error. The color scheme is consistent with the textual feedback which makes it easier to trace between the textual explanation and its graphical visualization. Sample generated textual and corresponding graphical feedback is presented in Figure 8. The architecture of the proposed realization model also allows the feedback generation package to be easily plugged in/out in the final output. The exception handler can serve as a (dis)connection gate.

3.5 Locating simulation feedback in the validation process

In terms of positioning the proposed feedback technique with respect to the modeling and semantic validation process, the following sequence is implied (see Figure 8): the user starts with analyzing a textual description of requirements. S/he will then transform the requirements into a conceptual model containing both the static and dynamic representations of a system. At any step during the modeling process the user can simulate the model by means of prototype generation. The prototype is then used to test a model in terms of its semantic conformance with the requirements. The model is revisited/refined if semantic errors are detected. The feedback is intended to facilitate the interpretation of the causes of the detected errors. Such repetitive trial/error loops will also allow to reflect on the requirements in terms of detection of ambiguous, missing or contradictory requirements.

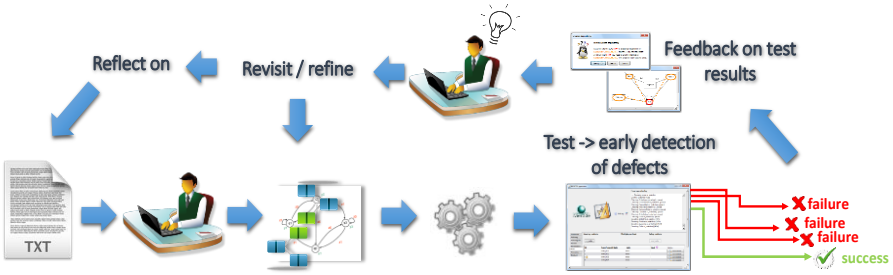

Figure 8: Positioning of the feedback in the modeling and validation process [49]

3.6 Assessing the effectiveness of feedback - enabled simulation as means for process-oriented cognitive feedback

An experimental study method was used to evaluate the feedback-enabled simulation with respect to learning effectiveness and usability. Six studies were conducted in the course of three academic years with participation of 201 master-level final year students from two Management Information Systems programs at KU Leuven. The effectiveness of the feedback-enabled simulation was assessed with respect to novices' comprehension of (i) structural aspects of a system represented as a class diagram [45], (ii) behavioral aspects of a system represented as multiple interacting statecharts [42], as well as understanding the interplay aspects between structural and behavioral views of a model [42] and (iii) hidden dependencies represented through inheritance hierarchies [44]. A classical pre/post-test control group experimental design was used in combination with a two-group and factorial designs [50]. During the experiments students had to validate a proposed model against given requirements by answering a set of questions (requirements reformulated as questions). The test results were scored in the range of min=0, max=8. The effectiveness of the proposed simulation method was measured by comparing students' test results between experimental cycles (without and with the use of simulation). A confirmatory analysis has been conducted to assess the validity of hypothesized effects. The results of the experimental studies showed a significant positive impact of the inclusion of the feedback on the semantic validation process of novices resulting in the average magnitude of effect of 2.33 out of 8 for validating the structural consistency (class diagram) [45], 4 out of 8 for validating the behavioral consistency (statecharts) and the consistency of behavioral aspects with the structural view of a system (contradicting constraints) [42], 2.33 out of 8 for validating hidden dependencies in a model represented through inheritance hierarchies [44]. This suggests that the proposed simulation method supports its intended goal as a ***cognitive feedback***.
Despite a tool's benefits, ***user acceptance*** however can be another important factor affecting its success. In the studies we chose to control important variables dealing with user acceptance. To test and evaluate the proposed design of the feedback-enabled simulation with respect to its subjective perceptions of usability by users (perceived easiness of use, perceived utility, preference and satisfaction) yearly evaluations were performed. Ease of use and usefulness are widespread and validated acceptance beliefs

from the Technology Acceptance Model [51-53], referring to the required effort to interact with a technology and its efficiency and effectiveness respectively. We used the concept of preference as another success dimension, as proposed by [54] and [55]. Preference is defined as "the positive and preferred choice for the continued use of simulation tool in the classroom". User satisfaction is another key success measure that has been defined as the feelings and attitudes that stem from aggregating all the efforts and benefits that an end user receives from using an system [56, 57]. Thereto a questionnaire was used including three questions per measurable dimension, each of which measured with a six-position Likert-type scale. Furthermore, context information about personal characteristics such as gender, previous knowledge, and the level of computer self-efficacy, was collected. Exploratory correlation analyses have been performed to study the correlation of the test results (relative advantage in score when using simulation) with user acceptance and personal characteristics. The findings from our analyses showed significant positive effects of the proposed feedback-enabled simulation on learning outcomes of novices regardless of personal characteristics and attitudes.

User acceptance of the feedback-enabled simulation tool was repeatedly evaluated in the course of several years of usage. The students found the tool useful and preferred its use (mean scores above 4.5 in six-position Likert-type scale). User satisfaction, preference, perceived usefulness and perceived ease of use were evaluated resulting respectively on average of 4.77, 4.78, 4.78 and 4.68 (with Cronbach Alpha above 0.84 and factor loadings per item above 0.86). The highest score in the anonymous evaluations was attributed by students to the incorporated feedback in the prototype (5.58 on average). Reliability and validity of the acceptance measures were assessed by factor analysis using SPSS. The findings from our analysis of acceptance variables show that, in addition, the students found the tool useful and preferred its continuous use during their learning process which suggests that the proposed simulation method supports its intended goal as a ***process-oriented feedback***. In addition, the use of CodeGen [8] during a learning process allows benefiting from the advantages of simulation-based learning by providing a learner with the opportunity to practice the obtained knowledge in order to obtain experience-based skills in the domain of conceptual modeling. As opposed to paper exercises which limit the scope of model understanding to a static view of a model, the dynamic testing with simulation fosters a more thorough understanding (cfr. challenge for teaching experience). In addition, the proposed method serves as a validation instrument allowing verifying the conformance of a model with the requirements (cfr. challenge for absence of validation tools). Using the insights from the testing a learner can either refine a model or reflect on the requirements by looking for instance for conflicting or missing requirements, allowing to improve the understanding of the domain to be engineered (cfr. challenge for lack in domain knowledge). The textual and visual feedback that is generated as a response to the errors during a testing process, allows linking the error with the causes in a model by also explaining the implications of the modeling constructs involved in the causes of the error (cfr. challenge for modeling language difficulties).

The reader is referred to [43-45, 58-61] for more details on these experimental evaluations.

4 Behavioral feedforward perspectives based on learning process analytics

While in the previous chapters cognitive feedback opportunities were investigated, in this chapter we discuss the perspectives for behavioral feedback based on learning process observation for modeling, i.e. feedforward opportunities that can reinforce a successful learning behavior.
Observing learning processes is however a challenging task considering the fact that learning is (meta)cognitive in nature. In order to observe learning processes several questions need to be answered:

- What is to be considered a learning process ?
- What type of data is needed to observe a learning process ?

In the context of this research we refer to the definition of a ***cognitive activity*** as "an operation that affects mental content, e.g. thinking, the cognitive operation of remembering, problem solving"; and to a *learning process* as "a composite cognitive activity that is concerned with acquisition of problem-solving abilities by which knowledge is acquired". In order to observe learning processes in the context of this research we make use of the traces produced during the ***problem-solving process*** of novices. We use the term "cognitive learning process" to refer to the set of modeling activities a learner performed within a modeling environment during his/her problem-solving process which are used as a proxy for the cognitive learning process [8]. During the semester students were assigned to a group project in which they were assigned the task of constructing a semantically correct conceptual model that reflects the structural and dynamic view of the given domain described in an approximately 5 page specification document based on real-world requirements. Modeling behavior data have been collected through the logging functionality of the MERODE modeling environment throughout a semester of study while students were working on their group's project. In order to observe the modeling process (how the novices created their models) interactions with the modeling tool have been logged conforming the *actor-event-target-timestamp*. As modeling manifests itself in the creation of modeling elements, in our logs we capture a modeling process as a sequence of create, edit, delete, undo, redo, and copy events. These events are further abstracted into CREATE and EDIT (grouping events edit, delete, undo, redo, copy) representations. For experimental data collection purposes the CodeGen simulation environment [8, 38, 40] was integrated within the MERODE modeling environment JMermaid allowing to, besides tracking only the modeling activities, log also simulation activities and thus observing the validation (also referred to as ***self-regulation***) activities within the task completion process.
Conceptual modeling event data of 36 cases (event logs of students' group works) from 2 academic years, 28.455 events in total have been subject to a three-dimension analysis using learning process analytics and ***process mining techniques*** in particular: 1. data has been examined at *different abstraction levels* (activities grouped for different modeling views such as structural or behavioral, fine-grained analysis zooming into each view in isolation), further expanded by diagram type information (class diagram –EDG,

statecharts –FSM, interaction view), event type information (e.g. create/edit/delete/simulate),element type (object, attribute, association, event, state, transitions, etc.) , 2. contrast analysis to identify differences based on *modeling performance* (best vs. worst scoring groups), 3. *time trend analysis* by making distinction between "early" and "late" sessions allowing to capture a change of behavior over time [9, 10].
The analysis resulted in identification of learning (i.e. modeling and simulation) behavior patterns indicative for worse/better learning outcomes. These first insights from our empirical studies suggest that the learning achievements, in addition to being related to cognitive aspects of learning, can be also associated with behavioral aspects [8-10] such as:

- Pattern 1 (modeling approach): best performance being associated with a more *iterative way* of modeling manifested in more frequent switches between different model views (such as structural and behavioral) as opposed to worst performance characterized by *sequential way* of working (targeting one task/view at a time).
- Pattern 2 (validation approach): best performance being characterized by *earlier engagement* n *validation (simulation)* activities with a *broader coverage of testing* targeting both structural and behavioral views of a model as opposed to worst performance characterized by limited coverage of model testing.
- Pattern 3 (validation target): best performance being characterized by the *intention to test recent changes*, and the validation effort being positioned around the *interplay effects between views* as opposed to worst performance characterized by a *general (unstructured) test and disconnected way of testing* targeting either structural or behavioral views.
- Pattern 4 (engagement styles): best performance being associated with an *earlier and systematic engagement* in modeling activities as opposed to worst performance characterized by *deadline-oriented engagement.*
- Pattern 5 (effort distribution across time): best performance being characterized by *more effort* put in the modeling process with a *tendency to decrease* over time as opposed to worst performance associated with *less effort* in earlier stages of project with a *tendency of continuous or increased effort* presumably indicating difficulties in achieving a right solution.
- Pattern 6 (effort distribution within modeling tasks): best performance being associated with a broader coverage of transforming requirements into a model in the early stages of the modeling process and continuing to adapt the model in later sessions, as opposed to worst performance associated with partial capturing of concepts and actively expanding the model in later sessions, by also supplying irrelevant concepts not required by requirements.

The findings showed that process analytics based feedback is feasible. Such a feedback can complement cognitive content related feedback (What is wrong and why ?) with a suggestive feedback targeting behavioral aspects, i.e. detecting inefficient learning processes and proposing recommendations on corrective actions (How to act?), i.e. feedforward. The findings are suggested as guidelines to improve teaching practices for multi-view conceptual modeling. The learning behavior patterns can also be used for construction of machine feedback targeting a modeling process in a learning context.

However, more research is needed towards 1. identification of more generic behavior patterns, 2. automation perspectives for learning behavior pattern detection that can be used to provide advanced real time individualized guidance throughout a learning process. The reader is referred to [8-10] for more details on these empirical studies.

5 Conclusion

The results of our research both for cognitive and behavioral aspects of learning suggest that validation (i.e. self-regulation using simulation) is positively associated with learning outcomes. MDE-based feedback-enabled simulation helps to improve knowledge of modeling concepts and modeling language by improving model understanding through ***reflecting on intermediate results (what is wrong ?)*** during a learning process (cfr. RA1). The findings of learning process analytics learning (modeling and validation) processes of novices establish the feasibility for feedback that can ***reflect on the procedural aspects of learning (how to do it the right way ?)*** thus complementing a cognitive feedback (cfr. RA2). While behavioral feedback based on the learning behavior patterns presented in our research would not be mature yet, however this research can serve as a platform to guide future research in the domain of learning process analytics and learning process analytics based feedback.
Two conclusions are obtained:

1. Simulation can serve a cognitive feedback throughout a learning process, if it is instant, easy to use and is easy to interpret (i.e. enhanced with a feedback that facilitates the interpretation of simulation results) (cfr. RA1).
2. Feedback perspectives based on learning process analytics are feasible. Process analytics (and process mining techniques in particular) make it possible to detect (in)efficient behavior during learning processes thus allowing to identify and address potential feedback needs earlier ***during a learning process*** (e.g. during a problem solving process) as opposed to learning ***process outcome feedback*** (provided only after a problem has been solved and its outcome is presented for assessment).

From a theoretical perspective the results of the first part of the research contribute to improving knowledge on the *cognitive aspects* of ***conceptual modeling*** providing empirical support for the use of simulation in learning/teaching processes for conceptual modeling with respect to supporting *model understandability* and thus also *model validity*. The results also contribute to the research on ***model-driven development*** with respect to its applicability to research on *simulation* and *feedback automation*. The research is also to be situated in the domain of ***simulation*** with respect to (1) empirical support for the use of augmented feedback in simulation, and (2) with respect to addressing the difficulties in interpretation of simulation results. Since our approach relies on process related data captured during a learning process, this study is also to be situated in the context of ***learning analytics***.
While the findings of the experiments showed a significant improvement in students' model-based validation capabilities when using feedback-enabled simulation, we still

observed certain issues. One issue is related to addressing the "completeness" dimension of a model's semantic quality. Since the completeness of a model can be demonstrated through testing scenarios, and the simulation only serves as instrument to execute the scenarios, transforming requirements into test scenarios is yet an additional skill that is required to benefit from the instrument. Our observations from the experiments revealed certain difficulties among students in developing testing scenarios for verifying their models with the use of simulation resulting in either (1) Omitted simulation cycle; or (2) Partial testing with the use of prototype characterized by incomplete testing scenarios such as a test scenario limited to only a confirmatory rather than exploratory analysis of the functionality, insufficient exploration of dependencies in a model, etc. [43, 58-61]. The observations of testing patterns of students thus suggest that combining the method of feedback-enabled simulation with the teaching of high level testing knowledge and skills will result in even better learning outcomes.

The main limitation with respect to our observations and analysis of behavioral aspects of learning include the missing perspectives on (1) individual learning processes since only group level information could be derived from the logs of the project file; (2) learning activities outside the learning environment (since learning is not limited to the scope of learning environments), which however would be very challenging to obtain.

The work presented in this paper can be expanded along several directions. Since the self-regulated activities (i.e. validation with the use of simulation) of novices were found to be key to distinguishing worse/better learning approaches, automated assistance can be investigated to provide tool support for (coverage) test scenario generation that will allow checking the "completeness" of a model with respect to the requirements. While findings showed that certain behavioral patterns can indeed be associated with better/worse outcomes in terms of reaching a satisfactory model quality, further examinations are needed to evolve towards more exhaustive and generic patterns for a broader learning context [62, 63]. Analysis of the testing logs from the simulation environment will provide more insights on (in)efficient testing processes which can be used to expand the simulation feedback ("What/why is not correct?") with feedforwarding possibilities during a modeling process (e.g. "When/what/how to test?"). Since learning processes are not limited to the scope of learning environments, correlating online with offline data (e.g. reasoning, perceiving, understanding, solving, reflecting, checking, …) can be another area of future research. The simulation and simulation feedback automation method proposed in this research can be extended to support a broader context of models, diagrams. Exploring perspectives of feedback personalization by means of studies at individual rather than group level can be another area of research. Advanced feedback mechanisms, can be explored using adaptive systems and learning reinforcement algorithms that also consider physiological indicators of learners, such as cognitive load, stress levels, affective states [64, 65], etc. Ultimately, the results our research can be inspirational beyond the scope of conceptual modeling.

References

1. Shute, V.J., *Focus on formative feedback.* Review of Educational Research, 2008. **78**(1): p. 153-189.

2. Nicol, D.J. and D. Macfarlane-Dick, *Formative assessment and self-regulated learning: a model and seven principles of good feedback practice.* Studies in Higher Education, 2006. **31**(2): p. 199-218.
3. Eyers, D., J. Jordan, and K. Hendry. *What are student perceptions of the timeliness of feedback?* . 2016 [cited 2016 April]; Available from: http://learning.cf.ac.uk/developing-educators/pcutl/project-reports/what-are-student-perceptions-of-the-timeliness-of-feedback/
4. Irons, A., *Enhancing learning through formative assessment and feedback*. 2007: Routledge.
5. Narciss, S., *Feedback strategies for interactive learning tasks.* Handbook of research on educational communications and technology, 2008: p. 125-144.
6. Butler, D.L. and P.H. Winne, *Feedback and Self-Regulated Learning: A Theoretical Synthesis.* Review of Educational Research, 1995. **65**(3): p. 245-281.
7. Zimmerman, B.J., *Investigating Self-Regulation and Motivation: Historical Background, Methodological Developments, and Future Prospects.* American Educational Research Journal, 2008. **45**(1): p. 166-183.
8. Sedrakyan, G., *Process-oriented feedback perspectives based on feedback-enabled simulation and learning process data analytics*. 2016, KU Leuven.
9. Sedrakyan, G., J. De Weerdt, and M. Snoeck, *Process-mining enabled feedback: "tell me what I did wrong" vs. "tell me how to do it right".* Computers in Human Behavior, 2016. **57**(C): p. 352-376.
10. Sedrakyan, G., M. Snoeck, and J. De Weerdt, *Process Mining Analysis of Conceptual Modeling Behavior of Novices - empirical study using JMermaid modeling and experimental logging environment.* Computers in Human Behavior, 2014. **41**(C): p. 486-503.
11. Schenk, K.D., N.P. Vitalari, and K.S. Davis, *Differences between Novice and Expert Systems Analysts: What Do We Know and What Do We Do?* Journal of Management Information Systems, 1998. **15**(1): p. 9-50.
12. Wang, W. and R.J. Brooks, *Empirical investigations of conceptual modeling and the modeling process.* Simulation Conference, 2007 Winter, 2007: p. 762-770.
13. Erickson, J. and S. Keng. *Can UML Be Simplified? Practitioner Use of UML in Separate Domains.* in *Proceedings of the 12th Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'07), held in conjunctiun with the 19th Conference on Advanced Information Systems (CAiSE'07),Trondheim, Norway*. 2007.
14. Wilmont, I., et al., *Cognitive Mechanisms of Conceptual Modelling*, in *Conceptual Modeling*, W. Ng, V. Storey, and J. Trujillo, Editors. 2013, Springer Berlin Heidelberg. p. 74-87.
15. Siau, K. and P.-P. Loo, *Identifying Difficulties in Learning Uml.* Information Systems Management, 2006. **23**(3): p. 43-51.
16. Shanks, G., E. Tansley, and R. Weber, *Using ontology to validate conceptual models.* Commun. ACM, 2003. **46**(10): p. 85-89.
17. Barjis, J., et al., *Innovative Teaching Using Simulation and Virtual Environments.* Interdisciplinary Journal of Information, Knowledge, and Management, 2012. **7**: p. 237-255.
18. Van Merriënboer, J.J. and P.A. Kirschner, *Ten steps to complex learning: A systematic approach to four-component instructional design*. 2012: Routledge.
19. Rutten, N., W.R. van Joolingen, and J.T. van der Veen, *The learning effects of computer simulations in science education.* Computers & Education, 2012. **58**(1): p. 136-153.
20. Akkoyun, O., and Nicola Careddu, *Mine simulation for educational purposes: A case study.* Computer Applications in Engineering Education, 2014.
21. Okutsu, M., DeLaurentis, D., Brophy, S., and Lambert, J. , *Teaching an aerospace engineering design course via virtual worlds: A comparative assessment of learning outcomes.* Computers & Education, 2013. **60**(1): p. 288-298.

22. Datta, A.K., V. Rakesh, and D. G. Way, *Simulation as an integrator in an undergraduate biological engineering curriculum.* Computer Applications in Engineering Education, 2013. **21**(4): p. 717-727.
23. Lateef, F., *Simulation-based learning: Just like the real thing.* Journal of emergencies, trauma, and shock, 2010. **3**(4): p. 348.
24. Gaba, D.M., *The future vision of simulation in healthcare.* Simulation in Healthcare, 2007. **2**(2): p. 126-135.
25. Lindland, O.I., G. Sindre, and A. Solvberg, *Understanding quality in conceptual modeling.* Software, IEEE, 1994. **11**(2): p. 42-49.
26. Nelson, H.J., et al., *A conceptual modeling quality framework.* Software Quality Journal, 2012. **20**(1): p. 201-228.
27. Lindland, O.I. and J. Krogstie. *Validating conceptual models by transformational prototyping*. in *Proceedings of the International Conference on Advanced Information Systems Engineering*. 1993. Springer.
28. Hess, T.A., *Investigation of Prototype Roles in Conceptual Design Using Case Study and Protocol Study Methods*. 2012, Clemson University.
29. Yang, M.C., *A study of prototypes, design activity, and design outcome.* Design Studies, 2005. **26**(6): p. 649-669.
30. Hevner, A., R., et al., *Design science in information systems research.* MIS Quarterly, 2004. **28**(1): p. 75-105.
31. Borland. *Keeping your business relevant with Model Driven Architecture (MDA)*. 2004; Available from: http://www.omg.org/mda/presentations.htm.
32. Gustas, R., *Conceptual modeling and integration of static and dynamic aspects of service architectures*, in *Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science*. 2010, Springer. p. 17-32.
33. Alanen, M. and I. Porres. *Model interchange using OMG standards*. in *Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on*. 2005. IEEE.
34. Desfray, P. *UML Profiles versus Metamodel extensions: An ongoing debate*. in *OMG's UML Workshops: UML in the. com Enterprise: Modeling CORBA, Components, XML/XMI and Metadata Workshop*. 2000.
35. Huang, S., V. Gohel, and S. Hsu. *Towards interoperability of UML tools for exchanging high-fidelity diagrams*. in *Proceedings of the 25th Annual ACM international Conference on Design of Communication*. 2007. ACM.
36. Lundell, B., et al., *UML model interchange in heterogeneous tool environments: an analysis of adoptions of XMI 2*, in *Model Driven Engineering Languages and Systems*. 2006, Springer. p. 619-630.
37. Snoeck, M., *Enterprise Information Systems Engineering: The MERODE Approach*. 2014: Springer.
38. Sedrakyan, G. and M. Snoeck. *A PIM-to-Code requirements engineering framework*. in *Proceedings of Modelsward 2013-1st International Conference on Model-driven Engineering and Software Development-Proceedings*. 2013.
39. Snoeck, M., et al., *Computer Aided Modelling Exercises.* Informatics in Education, 2007. **6**(1): p. 231-248.
40. Sedrakyan, G. and M. Snoeck, *Lightweight semantic prototyper for conceptual modeling*, in *Advances in Conceptual Modeling*. 2014, Springer. p. 298-302.
41. Prather, D.C., *Trial-and-error versus errorless learning: Training, transfer, and stress.* The American journal of psychology, 1971: p. 377-386.

42. Sedrakyan, G., S. Poelmans, and M. Snoeck, 2017. Assessing the influence of feedback-inclusive rapid prototyping on understanding the semantics of parallel UML statecharts by novice modellers. *Information and Software Technology*, *82*, pp.159-172.
43. Sedrakyan, G. and M. Snoeck. *Do we need to teach testing skills in courses on requirements engineering and modelling?* in *CEUR Workshop Proceedings*. 2014.
44. Sedrakyan, G. and M. Snoeck, *Effects of Simulation on Novices' Understanding of the Concept of Inheritance in Conceptual Modeling*, in *Advances in Conceptual Modeling*. 2015, Springer. p. 327-336.
45. Sedrakyan, G., M. Snoeck, and S. Poelmans, *Assessing the effectiveness of feedback enabled simulation in teaching conceptual modeling.* Computers & Education, 2014. **78**: p. 367-382.
46. Stefanidis, D., *Optimal acquisition and assessment of proficiency on simulators in surgery.* Surgical Clinics of North America, 2010. **90**(3): p. 475-489.
47. Ellis, R., *Corrective feedback and teacher development.* L2 Journal, 2009. **1**(1).
48. Ellis, R., *A typology of written corrective feedback types.* ELT journal, 2009. **63**(2): p. 97-107.
49. Sedrakyan, G. and M. Snoeck, *Enriching model execution with feedback to support testing of semantic conformance between models and requirements: Design and evaluation of feedback automation architecture*, in *Modelsward 2016 - 4th International Conference on Model-driven Engineering and Software Development*. 2016: Rome, Italy.
50. Trochim, W.M., *The Research Methods Knowledge Base, Internet WWW page.* http://trochim. human. cornell. edu/kb/index. htm. Version current as of August, 2000. **2**.
51. Davis, F.D., *Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology.* MIS Quarterly, 1989. **13**(3): p. 319-340.
52. Davis, F.D., R.P. Bagozzi, and P.R. Warshaw, *User acceptance of computer technology: a comparison of two theoretical models.* Management science, 1989. **35**(8): p. 982-1003.
53. Venkatesh, V., et al., *User acceptance of information technology: Toward a unified view.* MIS Quarterly, 2003. **27**(3).
54. Hsu, C.-L. and H.-P. Lu, *Consumer behavior in online game communities: A motivational factor perspective.* Computers in Human Behavior, 2007. **23**(3): p. 1642-1659.
55. Bourgonjon, J., et al., *Students' perceptions about the use of video games in the classroom.* Computers & Education, 2010. **54**(4): p. 1145-1156.
56. Ives, B., M.H. Olson, and J.J. Baroudi, *The measurement of user information satisfaction.* Communications of the ACM, 1983. **26**(10): p. 785-793.
57. Wixom, B.H. and P.A. Todd, *A theoretical integration of user satisfaction and technology acceptance.* Information systems research, 2005. **16**(1): p. 85-102.
58. Sedrakyan, G. and M. Snoeck, *Technology-enhanced support for learning conceptual modeling*, in *Enterprise, Business-Process and Information Systems Modeling*. 2012, Springer. p. 435-449.
59. Snoeck, M., and Sedrakyan, G. (2015). Tutorial: Boosting requirements analysis and validation skills through feedback-enabled semantic prototyping.
60. Snoeck, M., and Sedrakyan, G. (2015). Tutorial: Novel way of training conceptual modeling skills by means of feedback-enabled simulation.
61. Sedrakyan, G. and M. Snoeck, *Feedback-enabled MDA-prototyping effects on modeling knowledge*, in *Enterprise, Business-Process and Information Systems Modeling*. 2013, Springer. p. 411-425.
62. Sedrakyan, G., Järvelä, S., and Kirschner, P., Conceptual framework for feedback automation and personalization for designing learning analytics dashboards, 2016. Conference EARLI SIG 27, Online Measures of Learning Processes.

63. Sedrakyan, G., Malmberg, J., Noroozi, O., Verbert, K., Järvelä, S., and Kirschner, P., Designing a learning analytics dashboard for feedback to support learning regulation. submitted, 2017.
64. Sedrakyan, G., Leony, D., Munoz-Merino, P. J., Delgado Kloos, K. and Verbert, K., *Evaluating student-facing learning dashboards of affective states*, 12th European Conference on Technology Enhanced Learning (ECTEL'17) - Data Driven Approaches in Digital Education, Tallinn (Estonia), 2017.
65. Leony, D., Sedrakyan, G., Munoz-Merino, P. J., Delgado Kloos, K. and Verbert, K., Evaluating usability of affective state visualizations using AffectVis, an affect-aware dashboard for students, Journal of Research in Innovative Teaching & Learning, 2017.